VOLUME 22, NO.3 • MAY/JUNE 2001 (PUB-439)

BUILDING ENERGY SIMULATION

For Users of EnergyPlus, SPARK, DOE-2, BLAST, Genopt, Building Design Advisor, ENERGY-10 and their Derivatives

User News

What's New ?

....Release of EnergyPlus, Version 1.0

EnergyPlus 1.0 was released April 12. Get your free download today by visiting our website and clicking on "EnergyPlus 1.0" in the left menu:

http://SimulationResearch.lbl.gov

.....VisualSPARK 1.0

Purchase VisualSPARK 1.0. Information on p. 2.

.....New DOE-2 Consultant

We are happy to welcome **Kimberly Byk**, an energy simulation expert, as the newest DOE-2 consultant.

Wood, Byk & Associates, Inc.
829 Meadowview Road
Kennett Square, PA 19348
phone 610-347-0710
fax 610-347-0711
wba@utcorp.com

....DOE-2 Documentation on a CD

The Energy Science and Technology Software Center has scanned most of the DOE-2.1E documentation onto one CD. Cost is only $100; see p. 15 for ordering info. We are currently working in-house to convert the *DOE-2.1E Basics Manual* into pdf files for the web. Watch the *User News* for announcement of its availability.

...California's Energy Crisis at a Glance

Turn to p. 23 for this issue's featured web site: **http://energycrisis.lbl.gov**. The site provides a "snapshot" of the current supply and demand for electricity within California. The graph is updated every 10 minutes and you may watch the figure change by reloading the page.

What's Inside ?

EnergyPlus Version 1.0.1

A beta version of DOE's new EnergyPlus program with incremental improvements will be available this Fall. It will be designated as EnergyPlus 1.0.1. Watch the *User News* for announcement of its availability and list of new features.

Join the EnergyPlus User Group

The developers of EnergyPlus have formed a support group in order to foster discussion and maintain an archive of information for program users. We invite questions about program usage and suggestions for improvement to the code. This group is not meant to replace the primary support at EnergyPlus-Support@GARD.com.

The main page: http://groups.yahoo.com/group/EnergyPlus_Support
Send messages to: EnergyPlus_Support@yahoogroups.com

For information on EnergyPlus 1.0.0, or to download a free copy of the program, please go to

http://www.eren.doe.gov/buildings/energy_tools/energyplus

EnergyPlus is being developed by University of Illinois, CERL and Lawrence Berkeley National Laboratory, with the assistance of the Florida Solar Energy Center, GARD Analytics, Oklahoma State University, Krarti Associates, Pennsylvania State University, and the University of Wisconsin.

❾

VisualSPARK

Version 1.0

Available from Lawrence Berkeley National Laboratory,
VisualSPARK 1.0 allows you to build customized models of complex physical processes by connecting calculation objects. It is aimed at the simulation of innovative and/or complex building systems that are beyond the scope of programs like DOE-2 and EnergyPlus.

The main elements of VisualSPARK are a **user interface**, a **network specification language**, a **solver** for solving simultaneous algebraic and differential equations, and a **results processor**. With the network specification language you create equation-based calculation objects, and link the objects into networks that represent a building's envelope or HVAC components or systems. The solver solves this network for user-specified input parameters. With the results processor you graphically display the results of the calculation. VisualSPARK runs under the Windows 95/98/NT/2000, SunOS, Solaris, Linux and HPUNIX operating systems.

VisualSPARK costs $250. To purchase the program, go to
http://SimulationResearch.lbl.gov > VisualSPARK > Purchase

If you would like to get an idea of what the program does before purchasing it, you can review the SPARK User's Manual, which can be downloaded from http://SimulationResearch.lbl.gov > SPARK

VisualSPARK was developed by the LBNL Simulation Research Group and Ayres Sowell Associates, with support from the U.S. Department of Energy, Drury Crawley, program manager

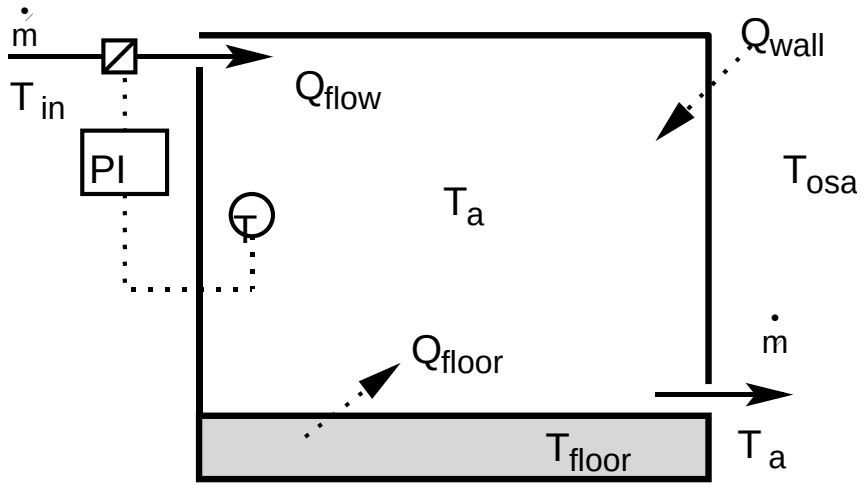
http://SimulationResearch.lbl.gov > SPARK

VisualSPARK 1.0: A Tutorial for the VisualSPARK GUI Part II*

**Brian V. Smith (bvsmith@lbl.gov), Dimitri Curtil (dcurtil@lbl.gov)
and Ender Erdem (aeerdem@lbl.gov)
Lawrence Berkeley National Laboratory**

Create a Proportional Integrating Controller and Supporting Classes

For this next example we will create a more complicated project that uses feedback to control the temperature in a room using a proportional integrating (PI) controller. The drawing ⇨ shows a schematic of the physical model of the room. Here is a system of equations for the single-zone room model:



$$\begin{cases} Q_{wall} = UA_{wall} \cdot \left(T_a - T_{osa}\right) \\ Q_{floor} = hA_{floor} \cdot \left(T_a - T_{floor}\right) \\ Q_{flow} = \dot{m}Cp \cdot \left(T_{in} - T_a\right) \\ Q_{floor} = Q_{flow} - Q_{wall} \\ cap_{floor} \cdot \dot{T}_{floor} = Q_{floor} \end{cases}$$

Equations for air cooler with PI controller:

$$\begin{cases} response_{PI} = K_P \cdot deviation + K_I \cdot \int_{t_0}^{t} deviation \cdot dt \\ response = \min\left(\max\left(response_{PI}, response_{\min}\right), response_{\max}\right) \end{cases}$$

Coupling equations between air cooler and room model:

$$\begin{cases} deviation = T_a - T_{set} \\ \dot{m}Cp = response \end{cases}$$

where:

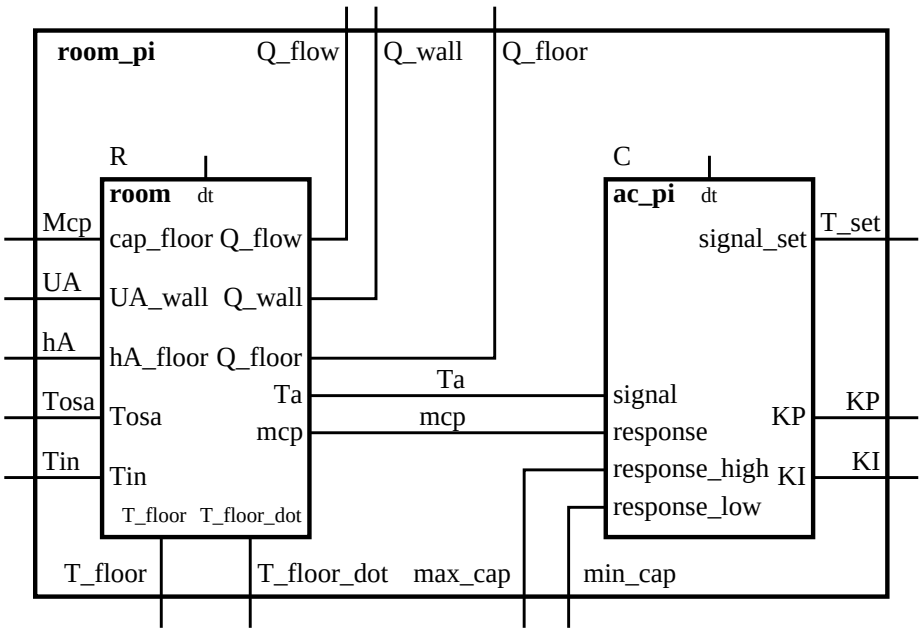
UA_{wall}	wall conductance [W/°C]
T_{osa}	outside air temperature [°C]
hA_{floor}	floor to room air conductance [W/°C]
T_{floor}	floor slab temperature [°C]
$\dot{T}_{floor}$	time-derivative of the floor temperature [°C/s]
T_a	room air temperature [°C]
T_{in}	supply air temperature [°C]
T_{set}	room set point temperature [°C]
Q_{wall}	heat flow from room air to walls and ceiling [W]

* Part I of this tutorial appeared in the January/February 2001 issue of the newsletter, Volume 22, Number 1

Q_{floor}	is the heat flow from room air to floor [W]
Q_{flow}	is the heat added (+) or removed (-) from the room due to air flow [W]
$\dot{m}Cp$	is the supply air capacity rate [W/°C]
cap_{floor}	is the floor slab heat capacity [J/°C]
$response_{min}$	is the minimum supply air capacity rate [W/°C]
$response_{max}$	is the maximum supply air capacity rate [W/°C],
K_P	is the controller's proportional gain [(W/°C)/°C]
K_I	is the controller's integral gain [(J/°C)/ °C]

Create the Project

Here is a schematic of the *SPARK* representation of the *room_pi* project. It shows various inputs and outputs and that two macro classes, *room* and *ac_pi* are used with some internal connections. The internal schematics of *room* and *ac_pi* will be shown later when we create their macro classes. Start *VisualSPARK* and click the **Project** menu button on the left column of buttons and select **New Project**. Enter the name **`room_pi`** in the dialog asking for the project name and press the **`<Enter>`** key.



When the editing panel pops up, copy and paste the following text into it:

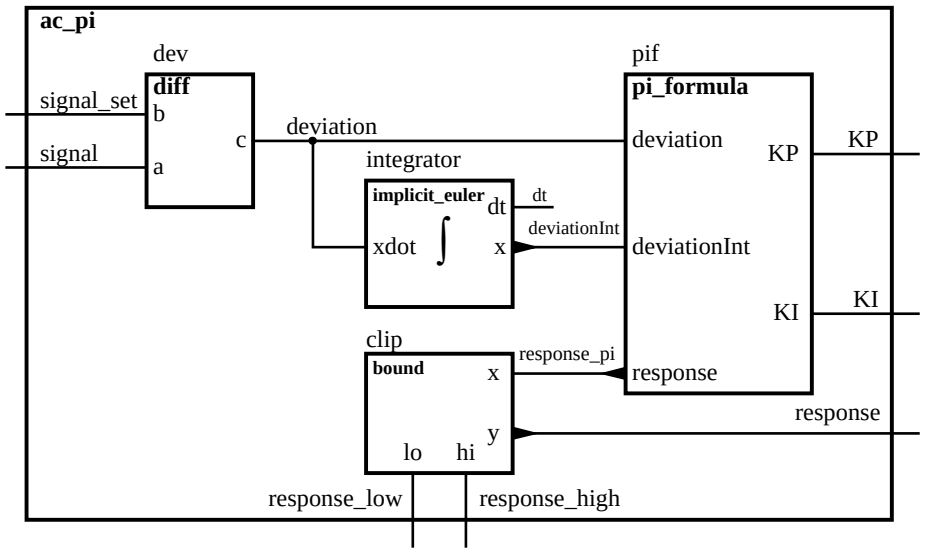
```
/*          room_pi.pr file
            Single zone room model with PI air temperature controller
*/

DECLARE  ac_pi     C;  // Air cooler with PI controller
DECLARE  room      R;  // Single zone room model

// Inputs for AC with PI controller
INPUT KP           C.KP;
INPUT KI           C.KI;
INPUT T_set        C.signal_set                INIT=20.0   REPORT;
INPUT max_cap      C.response_high [W/deg_C]   INIT=100.0  REPORT;
INPUT min_cap      C.response_low  [W/deg_C]   INIT=0.0    REPORT;

// Inputs for single zone room model
INPUT UA_wall      R.UA_wall     [W/deg_c];
INPUT cap_floor    R.cap_floor   [J/deg_C];
INPUT hA_floor     R.hA_floor    [W/deg_C];
INPUT Tosa         R.Tosa        [deg_C]                   REPORT;
INPUT Tin          R.Tin         [deg_C]                   REPORT;

// Heat transfers
LINK Q_flow        R.Q_flow      [W]                       REPORT;
LINK Q_wall        R.Q_wall      [W]                       REPORT;
```

```
LINK Q_floor        R.Q_floor      [W]                          REPORT;

// Floor temperature
LINK T_floor        R.T_floor      [deg_C]         INIT=30.0    REPORT;
LINK T_floor_dot    R.T_floor_dot  [deg_C/s]                    REPORT;

// Supply air capacity rate
LINK mcp            R.mcp,
                    C.response     [W/deg_C]                    REPORT;

// Room air temperature
LINK Ta             R.Ta,
                    C.signal       [deg_C]         INIT=20.0    REPORT;
```

To do this on Microsoft Windows platforms, select the text with the mouse, press Control-C, click on the *VisualSPARK* edit panel and press Control-V. On Unix platforms, select the text with the mouse, click on the *VisualSPARK* edit panel and press Control-Y (yank). You may first have to turn on *text select mode* in your PDF reader. Next, click the **Save** button (the floppy disk icon) followed by the **Close** button. In the Projects area in the main *VisualSPARK* window, click on the line that contains *room_pi*. Because the *ac_pi* and *room* macro classes aren't yet defined, there will be errors. Two windows will appear – a dialog box complaining that there is no makefile.inc and a larger window titled *VisualSPARK – ERROR, Showing: parser.log – parser.log* with text saying that it can't find class *ac_pi*. Just click on the **OK** button in the small dialog box and the **Close** button in the larger error window.

Create the Macro Class ac_pi and the Atomic Class pi_formula

Here is a schematic representation of the *ac_pi* controller macro class that shows four atomic classes, *diff, bound, implicit_euler* and *pi_formula*. The *diff* class calculates the difference between two values, *signal* (a) and *signal_set* (b), producing *deviation* (c). The atomic class *implicit_euler* is an integrator, which integrates *deviation* (xdot) producing *deviationInt* (x). The atomic class *bound* bounds a value by two extremes, the *hi* and *lo* values, where *lo* must be smaller than *hi*.



diff, clip and *implicit_euler* are defined in the globalclass directory. Outgoing arrows indicate that the atomic class provides an inverse only for the port with the arrow, thus forcing the computational flow in the direction of the arrow. We will be creating the *pi_formula* atomic class later, where it will also be described. In the *Class Directories* area in the *VisualSPARK* window click on the entry that contains "**.** (room_pi)", then click the **New Class** menu and select **Create macro class**. Enter the name **`ac_pi`** for the macro class into the dialog followed by the <**`Enter`**> key. When the editing panel pops up, erase the text already in the window and cut and paste the following text into it:

```
/*                Air cooler with PI controller
 *                     Macro
 *                   ac_pi.cm
```

```
 */

PORT response;
PORT response_low;
PORT response_high;

PORT signal;
PORT signal_set;

PORT KP;
PORT KI;

DECLARE  diff           dev;        // To compute deviation = signal - signal_set
DECLARE  pi_formula     pif;        // PI controller's formula that computes the
                                    // controller's response
DECLARE  implicit_euler integrator; // Note: all integrators used in the problem
                                    // should be the same.
DECLARE  bound          clip;       // The controller's response must lie between
                                    // response_low and response_high.

LINK .KP                pif.KP;
LINK .KI                pif.KI;

// PI formula
LINK .response          clip.y;
LINK response_pi        pif.response,
                        clip.x                          REPORT;
LINK .response_low      clip.lo;
LINK .response_high     clip.hi;

// deviation = signal - signal_set
LINK .signal_set        dev.b;
LINK .signal            dev.a;
LINK deviation          dev.c,
                        pif.deviation,
                        integrator.xdot                 REPORT;

// Time integral of the deviation
LINK deviationInt       integrator.x,
                        pif.deviationInt                REPORT;

LINK dt                 integrator.dt                   GLOBAL_TIME_STEP;
```

Now click the **Save** button (the floppy disk icon) followed by the **Close** button. At this point, the *Classes* area in the main VisualSPARK window will contain the macro class file `ac_pi.cm`.

Next we will create the *pi_formula* atomic class that is used in the *ac_pi* macro class. This class multiplies *deviation* by *KP* and adds that to the product of *deviationInt* and *KI*, producing *response*. Make sure the entry "**.** (room_pi)" is still selected in the main window and again click the **New Class** menu and select **Create atomic class**. Enter the name **`pi_formula`** in the dialog followed by the **`<Enter>`** key. When the editing panel pops up, erase the text already in the window and cut and paste the following text into it:

```
/*              PI controller
 *              Atomic class : pi_formula.cc
 */

#ifdef SPARK_PARSER
```

```
// Force computational flow by providing only one inverse (equivalent to
MATCH_LEVEL=10)
PORT response     "controller's response"                MATCH_LEVEL=10;

PORT deviation    "deviation from set value";
PORT deviationInt "integrated deviation from set value"   INIT=0.0;

PORT KP           "parameter for the proportional part";
PORT KI           "parameter for the integrating part";

FUNCTIONS {
   response = pi_formula(KP, KI, deviation, deviationInt);
}

#endif /* SPARK_PARSER */

#include "spark.h"

////////////////////////////////////////////////////////
// Function name   : pi_formula
// Description     : Implements PI controller formula with
//                   special treatment for the initial time solution
//
//      Controller is not ON for the initial time solution: inverse returns
response=0.0
//      when the global function ::IsInitialTime() returns true.
//      This allows us to compute the correct physical state of the uncontrolled
//      system at InitialTime.
//      Otherwise the initial solution would depend on the values of the
//      KP, KI and deviationInt variables.
//
//      Also, we must enforce deviationInt = 0 at InitialTime.
//      See corresponding PORT declaration with INIT=0.0
//
// Return type     : double
// Argument        : ArgList args
double pi_formula(ArgList args)
{
   const double KP = args[0];
   const double KI = args[1];
   const double deviation = args[2];
   const double deviationInt = args[3];
   double result;

   if ( ::IsInitialTime() ) { // Initial time solution only
      result = 0.0;
   }
   else { // Used after initial time solution: PI controller formula
      result = KP*deviation + KI*deviationInt;
   }
   return result;
}
////////////////////////////////////////////////////////
```

For the initial time solution, the controller's response is set to zero so that the controller does not impact the initial state of the physical system being controlled. Otherwise, the value of the room air temperature *Ta* at the initial time would depend on the controller's gains *KP* and *KI*. This special treatment for the initial time solution is

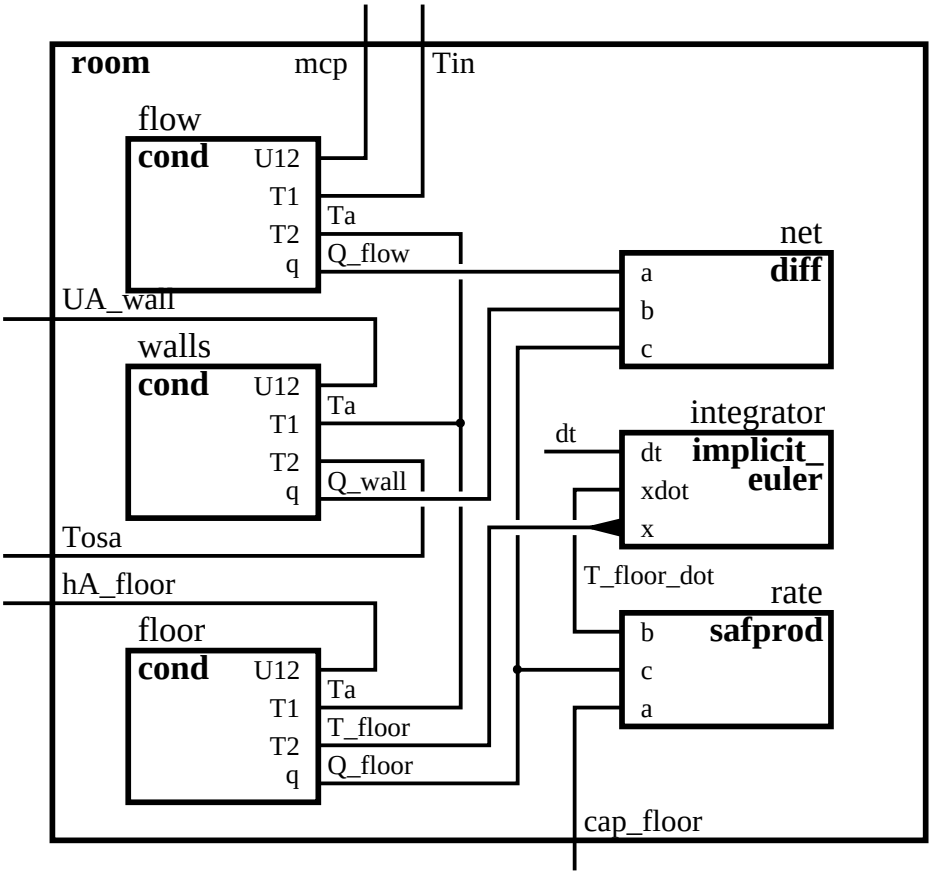
implemented using the global predicate function **`IsInitialTime()`** that returns true only when the global time is equal to the initial time specified in the run-control file.

Note that the integral *deviationInt* of the variable *deviation* is initialized to zero. The idea is that every time the controller is turned on (i.e., starts operating), the variable *deviationInt* should be reset to zero so that the controller operates correctly. However, since there is no mechanism in *SPARK* to reset a dynamic variable to some "initial" value after the initial time, the current model can only be used when the controller is switched on just after the initial time solution. Note that there is no such initialization problem with a P-controller. Now click the **Save** button (the floppy disk icon) followed by the **Close** button. At this point, the *Classes* area in the main *VisualSPARK* window will contain the macro class file `ac_pi.cm` and the atomic class file `pi_formula.cc`.

Create the Macro Class room

The last class we need to create is *room* macro class. Here is a schematic of *room* that uses four atomic classes:

1. The *cond* class calculates heat flow as a function of conductance (U-value) and Temperature difference as follows: $Q = U12 \times (T1 - T2)$, where Q is the heat flow, $U12$ is UA (i.e., conductance x area) or $\dot{m}Cp$ (i.e., mass flow x heat capacity), and $T1$ and $T2$ are temperatures.
2. The *diff* class calculates the difference of two variables. The equation is $c = a - b$.
3. The *implicit_euler* class implements the implicit Euler integration scheme, which integrates *xdot* with respect to *dt*, resulting in *x*.
4. The *safprod* class calculates the product of two variables, but has a "safe" inverse that returns a very large number when dividing by 0. The equation is $c = a \times b$.



With the *Class Directory* "**.** (room_pi)" still selected in the main window, click the **New Class** menu and select **Create macro class**. Enter the name **`room`** in the dialog followed by the **`<Enter>`** key. When the editing panel pops up, erase the text already in the window and cut and paste the following text into it:

```
/*                Massive Floor Room
 *                     Macro
 *                   room.cm
 */

// Temperatures
PORT  Ta          [deg_C]   "Room air temperature";
PORT  T_floor     [deg_C]   "Room floor temperature";
PORT  T_floor_dot [deg_C/s] "Room floor temperature rate of change";
```

```
PORT  Tosa        [deg_C]   "Outside air temperature";
PORT  Tin         [deg_C]   "Supply air temperature";

// Conductances and heat capacities
PORT  UA_wall     [W/deg_C] "Wall conductance";
PORT  hA_floor    [W/deg_C] "Floor to air conductance";
PORT  cap_floor   [J/deg_C] "Floor mass heat capacity";
PORT  mcp         [W/deg_C] "Supply air heat capacity rate";

// Heat transfers
PORT  Q_flow      [W]       "Heat added (+) or removed (-) by air stream";
PORT  Q_wall      [W]       "Wall heat transfer";
PORT  Q_floor     [W]       "Heat from air to floor";

DECLARE  cond           flow;      //  Air mass flow "conduction"
DECLARE  cond           walls;     //  Wall conduction
DECLARE  cond           floor;     //  Floor to air conduction
DECLARE  diff           net;       //  Diff between Q in and Q out
DECLARE  safprod        rate;      //  Multiply T_floor_dot* Mcp
DECLARE  implicit_euler integrator;//  Implicit Euler integrator

LINK  .Tosa,            walls.T2;
LINK  .Tin,             flow.T1;
LINK  .UA_wall,         walls.U12;
LINK  .hA_floor,        floor.U12;
LINK  .mcp,             flow.U12;
LINK  .cap_floor,       rate.a;
LINK  .Q_wall,          walls.q,
                        net.b;
LINK  .T_floor,         floor.T2,
                        integrator.x;
LINK  .T_floor_dot,     rate.b,
                        integrator.xdot;
LINK  .Q_floor,         floor.q,
                        net.c,
                        rate.c;
LINK  .Ta,              flow.T2,
                        walls.T1,
                        floor.T1;
LINK  .Q_flow,          flow.q,
                        net.a;

LINK  dt,               integrator.dt       GLOBAL_TIME_STEP;
```

Now click the **Save** button (the floppy disk icon) followed by the **Close** button. At this point, the *Classes* area in the main *VisualSPARK* window should contain the macro class file room.cm along with the previously created ac_pi.cm and pi_formula.cc files. Since the classes *implicit_euler*, *diff* and *safprod* are defined in the globalclass directory and the class *cond* is defined in the `hvactk` class directory, they do not need to be created.

Create a New Input Set and Run the Problem

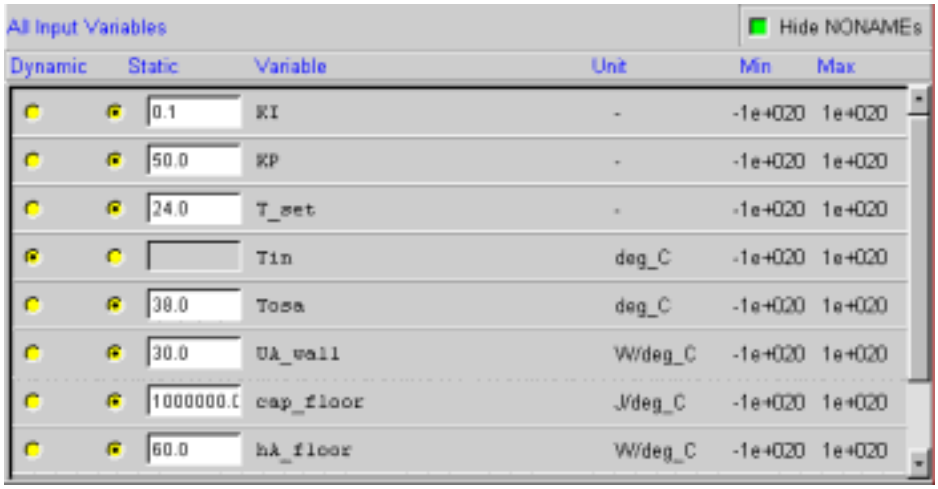
In the main VisualSPARK window select the line that contains *room_pi* in the *Projects* area. Now click on the **New Input Set** button in the left column of buttons in the main window. This will bring up a dialog where you may enter a name for the input data set. Type **`input1`** there and press **`<Enter>`**.

A panel will pop up where you may enter input data for the problem. This is called the input panel. Some of the variables are static, meaning that they don't change during the run of the problem, and one is dynamic, with different values defined at specific time stamps.

Specify Static Input Values

The middle area of the input panel labeled *All Input Variables* shows the variables that are used in the problem. For the static variables, click on the check button under the column labeled **Static** and enter the value in the box to its right. The following figure, at left, is what you should enter for each static variable (design parameters). The supply air capacity rate *mcp* must be positive, hence *min_cap* = 0. We further constrain the controller's response by requiring that the supply air capacity rate be smaller than *max_cap*. The input variable area is scrollable so if you don't see all the variables just scroll down using the scrollbar on the right side. The figure below is what you should see at this point.

KI		0.1
KP		50
cap_floor		1.0e6
T_set		24
Tosa		38
UA_wall		30
hA_floor		60
min_cap		0
max_cap		100



Specify Dynamic Input Values

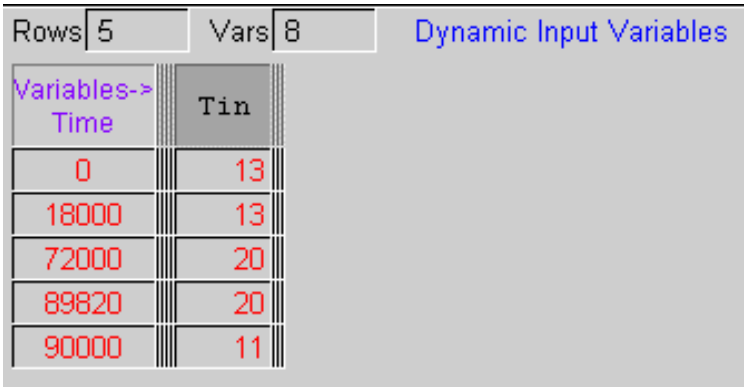
In the bottom section labeled *Dynamic Input Variables* is a table (grid) where values may be entered for the dynamic input variables. You may have noticed that originally all the input variables were displayed in that table, but as you checked the **Static** check button for each variable, they disappeared from the *Dynamic Input Variables* table. At this time you should only see the *Tin* variable. Also notice that there isn't much space showing for this table. You can change that by clicking on the small square on the far right separating the upper and lower halves of the input panel and dragging it up, making the lower half taller and the upper half shorter. You may also resize the whole input panel in the usual method of resizing windows on your system.

At the very bottom of the input panel you will see three buttons – **Insert Row**, **Add Row** and **Delete Row**. These control the rows in the dynamic input variable section. The difference between **Insert Row** and **Add Row** is that the former inserts a row in the table *before* the selected row and the latter adds a row *after* it. If no row is selected, both **Insert Row** and **Add Row** insert a row *before* row 0 in the table.

Now, click **Insert Row** five times to make five rows and enter the following data for the time and temperatures for the *Tin* variable ⇩

Time	**Tin**
0	13
18000	13
72000	20
89820	20
90000	11

⇨



If the model runs beyond 90000 seconds, the last value of *Tin* (11) will be used. Here is what the table should look like at this point.

Specify Run-Time Parameters

Finally, we will specify the run-time parameters for the model. This tells it when to start, when to stop, the time increment, and when and how often to report results. It is also here that you may specify a diagnostic level for debugging information.

Enter the values in the far-right area of the input panel

Finally, click the **Save** button (the floppy disk icon).

Specify Initial Values for Dynamic Variables and Break Variables

Other variables besides the input variables require initial values to be specified. Every dynamic variable, i.e., a variable that is connected to the *x* port of an integrator object, needs an initial value. Initial values for unknown variables cannot be specified using the *VisualSPARK* input editor in version 1.0.1. The language construct **`INIT=initial_value`** must be used in the description of the *SPARK* problem, either in a **`LINK`** statement in the problem file or a macro class, or in a **`PORT`** statement in an atomic class.

In the *room_pi* problem there are two dynamic variables, namely *T_floor*, defined in the macro class *room*, and *deviationInt*, defined in the class *pi_formula*. You should make sure that initial values for these two variables are correctly specified using the **`INIT`** keyword, either in the relevant classes or in the problem (`.pr`) file.

We defined the following initial values for the dynamic variables:

`LINK T_floor ...` **`INIT = 30.0 ...;`** in the file `room_pi.pr`,

and

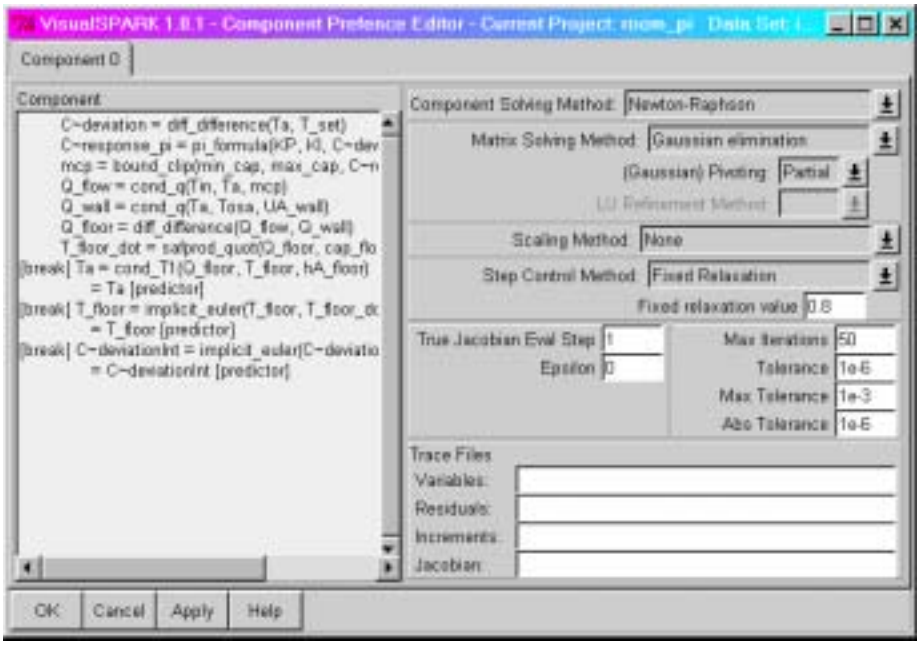
`PORT deviationInt ...` **`INIT = 0.0 ...;`** in the file `pi_formula.cc`

In addition to initial values, predictor values can be specified for the break variables that are not dynamic variables. In order to see which are the break variables in our problem, you need to check the preferences for the components constituting the *room_pi* problem. To do this, go back to the main *VisualSPARK* window and clicking the **Preferences** button after making sure that the *input1* data set is still selected under the *Projects* area. This will pop up a window called the Component Preference Editor. There will be a tab for each component comprising the *room_pi* problem. In this problem there is only one component, named *Component 0*.

In the text area under the tab you will see the solution sequence describing the component. The break variables are tagged with the keyword **`[break].`** In addition to the two dynamic variables discussed above, there is another break variable, namely the room air temperature, *Ta*. We defined an initial predictor value for this break variable using the **`INIT`** construct in the declaration of the link *Ta* in the problem file `room_pi.pr`:

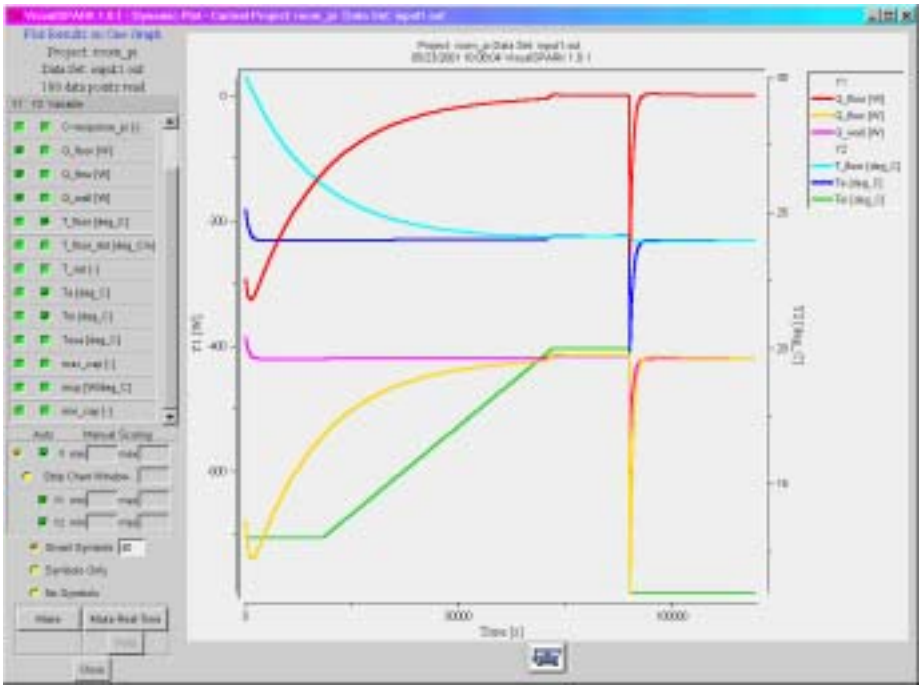
```
LINK Ta ... INIT = 20.0 ...;
```

Note that it is not compulsory to specify an initial predictor value for a break variable that is not a dynamic variable, but it usually speeds up the computation of the initial time solution if the specified initial predictors are relatively close to the solution. Now enter a value of `0.8` for the **Relaxation Coefficient**. If the default value of `1.0` is used, the solver will fail to converge after 180 seconds and the simulation will stop at that point. The others should default to the correct settings. See Section 3.10.3 in the *SPARK User's Manual* for more details on the usage of the relaxation coefficient. The panel should look like this ⇨



Run Simulation and Plot the Results

In the main *VisualSPARK* window click the **Run** button just below the **New Input Set** button. A *Run Status* window will pop up showing the progress of the compilation and run of the simulation. When the simulation window finishes, the *Run Status* window closes. At any time during the compilation or simulation run you can click the **Stop** button to abort the process. Now click on the **Results/Plots** menu button below the **Run** button and select **Dynamic, multiple variables per plot**, as we want to plot several variables on the same graph. A dialog will pop up asking for the data file name.



Double click on `input1.out` to select the output file that *SPARK* just created. Here is what a typical graph should look like for the *room_pi* problem. As a further exercise, you might try entering time-varying values for *Tosa* and/or *T_set* and see how they affect the controller's operation. Also, you could try different values for *max_cap* to see the effect on the system's behavior.

Remember to save any changes you make in the input data panel before you make a run.

Recent Reports

LBNL-48371

GenOpt® – A Generic Optimization Program

Michael Wetter
Simulation Research Group
Lawrence Berkeley National Laboratory

The potential offered by computer simulation is often not realized: Due to the interaction of system variables, simulation users rarely know how to choose input parameter settings that lead to optimal performance of a given system. Thus, a program called GenOpt r that automatically determines optimal parameter settings has been developed. GenOpt is a generic optimization program; it minimizes an objective function with respect to multiple parameters. The objective function is evaluated by a simulation program that is iteratively called by GenOpt. In thermal building simulation – which is the main target of GenOpt – the simulation program usually has text-based I/O. The paper shows how GenOpt's simulation program interface allows the coupling of any simulation program with text based I/O by simply editing a configuration file, avoiding code modification of the simulation program. By using object-oriented programming, a high-level interface for adding minimization algorithms to GenOpt's library has been developed. We show how the algorithm interface separates the minimization algorithms and GenOpt's kernel, which allows implementing additional algorithms without being familiar with the kernel or having to recompile it. The algorithms can access utility classes that are commonly used for minimization, such as optimality check, line-search, etc. GenOpt has successfully solved various optimization problems in thermal building simulation. We show an example of minimizing source energy consumption of an office building using EnergyPlus, and of minimizing auxiliary electric energy of a solar domestic hot water system using TRNSYS. For both examples, the time required to set up the optimization was less than one hour, and the energy savings are about 15%, together with better daylighting usage or lower investment costs, respectively.

This paper will be presented at the "Building Simulation 2001" Conference, Rio de Janeiro, August 13-15, 2001, and will be published in the Proceedings.

❾

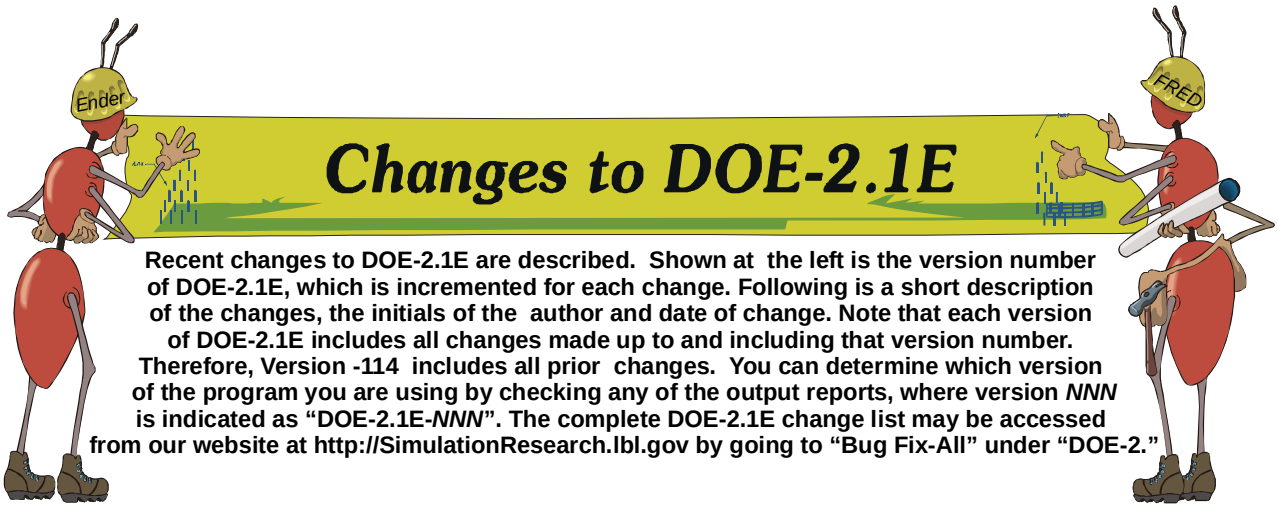
GenOpt® 1.1

New in GenOpt 1.1 are an additional algorithm for multi-dimensional optimization, algorithms for one-dimensional optimization, and an algorithm for parametric runs in a multi-dimensional space. The new version also allows processing of multiple function values and has an improved graphical user interface.

GenOpt is a multi-parameter optimization program, available free of charge from LBNL. It automatically finds the values of user-selected design parameters that minimize an *objective function*, such as annual energy use, calculated by an external simulation program like EnergyPlus, SPARK, DOE-2, BLAST, TRACE, TRNSYS, etc. GenOpt can be used with any simulation program that has text-based input and output. It also offers an interface for adding custom optimization algorithms to its library.

Genopt 1.1 (with user manual) may be downloaded from

http://SimulationResearch.lbl.gov > GenOpt



Changes to DOE-2.1E

Recent changes to DOE-2.1E are described. Shown at the left is the version number of DOE-2.1E, which is incremented for each change. Following is a short description of the changes, the initials of the author and date of change. Note that each version of DOE-2.1E includes all changes made up to and including that version number. Therefore, Version -114 includes all prior changes. You can determine which version of the program you are using by checking any of the output reports, where version *NNN* is indicated as "DOE-2.1E-*NNN*". The complete DOE-2.1E change list may be accessed from our website at http://SimulationResearch.lbl.gov by going to "Bug Fix-All" under "DOE-2."

-112 : dkey sys

1. The FROM-GROUND code-word (in the HP-LOOP-HEATING keyword under the PLANT-ASSIGNMENT command) was enabled: e.g. HP-LOOP-HEATING=FROM-GROUND is allowed. [EE 2000.10.05]

2. Some compiler warnings in SYSTEMS were fixed. [EE 2000.10.02]

-113 : bdl dkey sim

1. In doebdl the memory was increased to 900000. [EE 2001.04.10]

2. The limits of these commands were increased: [EE 2001.04.10]

Command Name	old limit	new limit
MATERIAL	128	1024
CONSTRUCTION	64	128
POLYGON	4000	5000
EXTERIOR-WALL	300	2048
WINDOW	200	2048
INTERIOR-WALL	512	2048
SYSTEM	100	128

3. In doesim the memory was increased to 900000. [EE 2001.04.10]

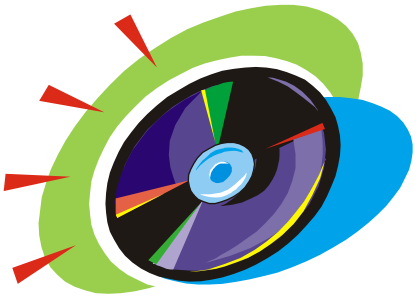
-114 : sys [WFB 2001.04.27]

In Systems in the water-loop heat pump system (SYSTEM-TYPE = HP) the option HP-LOOP-HEATING = FROM-PLANT does not work. This option passes the loop heating load on to Plant instead of simulating a boiler in Systems. The loop temperature is always getting set to zero resulting in incorrect very large loads getting sent to Plant. This change fixes the problem. There is no work around.

DOE-2 Documentation on a CD !

What's on the CD?

- DOE-2 Reference Manual (Part 1)
- DOE-2 Reference Manual (Part 2)
- DOE-2 Supplement to the Reference Manual (2.1E)
- DOE-2 BDL Summary (2.1E)
- DOE-2 Engineers Manual (2.1A)



How much does it Cost?

- Cost of the CD is U.S.$100.

Order from ESTSC:

Ed Kidd
NCI Information Systems, Inc.
Energy Science and Technology Software Center
P.O. Box 1020
Oak Ridge, TN 37831

Phone: 865/576-1037
Fax: 865/576-6436
Email: estsc@adonis.osti.gov

What Isn't on the CD?	Where to Obtain Printed Documentation:
▪ Update Package #1: Changes and corrections to DOE-2.1E Basics, the Supplement and BDL Summary ▪ Update Package #2: Corrections to the BDL Summary and Supplement for DOE-2.1E. For Version 107 of DOE-2.1E, added Cooled Beam System and Polygon sections to the Supplement and BDL Summary. ▪ Update Package #3: Corrections to Appendix A of the Supplement.	Update Packages are pdf files; they may be downloaded from our website at http://SimulationResearch.lbl.gov > DOE-2 > Documentation Update Packages are **not** cumulative and each contains different information. You must download all three packages to update the DOE-2 documentation completely.
▪ DOE-2 Basics (2.1E) ▪ DOE-2 Sample Run Book (2.1E)	These must be purchased separately from NTIS; details at http://SimulationResearch.lbl.gov > DOE-2 > Documentation]

Disclaimer: The Building Energy Simulation User News was prepared as an account of work sponsored by the United States Government (USG). While this document is believed to contain correct information, neither the USG nor any agency thereof, nor the Regents of the University of California (RUC), nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process or service by its trade name, trademark, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the USG or any agency thereof, or the RUC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the USG or any agency thereof or of the Regents of the University of California

ENERGY-10, Version 1.3 with WeatherMaker

Version 1.3 of ENERGY-10 is now available. It includes the much-anticipated ***WeatherMaker*** function. *WeatherMaker* allows users to create their own weather files based on information available from nearly 4,000 weather stations throughout the U.S. Revisions to the program itself include some minor fixes, an improved and expanded Help section, and greater clarity in titling and identification of various sections. Contact the Sustainable Buildings Industries Council for more information, or to order your upgrade disc (the cost is $15, which covers production and shipping).

ENERGY-10, written in C^{++}, is a design tool for smaller residential or commercial buildings that are less than 10,000 ft^2 floor area, or buildings that can be treated as one- or two-zone increments. It performs whole-building energy analysis for 8760 hours/year, including dynamic thermal and daylighting calculations. ENERGY-10 was specifically designed to facilitate the evaluation of energy-efficient building features in the very early stages of the design process.

Input: Only four inputs required to generate two initial generic building descriptions. Virtually everything is defaulted but modifiable. As the design evolves, the user adjusts descriptions using fill-in menus (utility-rate schedules, construction details, materials).

Output: Summary table and 20 graphical outputs available, generally comparing current design with base case. Detailed tabular results also available.

Platform: PC-compatible, Windows 3.1/95/98, Pentium processor with 16 MB of RAM is recommended.

Douglas K. Schroeder
1331 H Street N.W., #1000
Washington, DC 20004

Tel: 202.628.7400 ext 210
Fax: 202.383.5043
www.sbicouncil.org

Sustainable Buildings Industry Council (SBIC)

San Diego Gas & Electric

Whole Building Performance Training

SDGE® A Sempra Energy® company

REGISTER AT HTTP://WWW2.SDGE.COM/SEMINAR

August 03 (Friday) 9:00 am to 3:00 pm	**Design Underfloor Air Distribution Systems for Maximum Efficiency**
September 13 (Thursday) 8:30 am to 4:00 pm	**H-P Design Strategies: Lighting, Windows and Building Envelopes with EnergyPro 3.0***
September 14 (Friday) 8:30 am to 11:30 am	**Mechanical System Design and Modeling Using EnergyPro 3.0***
September 14 (Friday) 1:00 pm to 4:30 pm	**Advanced Building Modeling with EnergyPro 3.0***

*** See page 28 for EnergyPro 3.0 information**

Building Design Advisor 3.0

Decision making through the integrated use of multiple simulation tools and databases

The **Building Design Advisor (BDA)** is a Windows program that addresses the needs of building decision-makers from the initial, schematic phases of building design through the detailed specification of building components and systems. The BDA is built around an object-oriented representation of the building and its context, which is mapped onto the corresponding representations of multiple tools and databases. It then acts as a ***data manager*** and ***process controller***, automatically preparing input to simulation tools and integrating their output in ways that support multi-criterion decision-making. Version 3.0 of the BDA is now available for Beta testing and includes links to three main simulation tools for daylighting, electric lighting and energy analyses:

- **DCM**, a simplified daylighting simulation tool,
- **ECM**, a simplified electric lighting simulation tool, and
- the **DOE-2.1E** building energy simulation program.

ECM, the **new electric lighting simulation tool** in BDA 3.0, is integrated through BDA with DOE-2. BDA's Schematic Graphic Editor allows placement of electric lighting luminaires and specification of reference points for daylight-based electric lighting controls. Moreover, BDA now has the capability of **running DOE-2 parametrically** to generate a plot that shows the relationship between effective aperture and energy requirements. BDA 3.0 provides the added functionality of working with either **English units or Metric units**.

Current research and development efforts are focused on the development of links to **Desktop Radiance**, a Windows 95/98/NT version of the **Radiance** lighting/daylighting simulation and rendering software.

The minimum and recommended system **requirements** to run the BDA software are as follows:

Minimum	**Recommended**
Pentium 75	Pentium 200 or better.
Windows 95, 98, NT 4.0.	Windows 95, 98, NT 4.0.
16 / 32MB RAM under Windows 95	24 / 64MB RAM under Windows NT 4.0.
30 MB of larger hard disk space.	60 MB of larger hard disk space.
640x480 or higher screen resolution.	1024x768 or higher screen resolution.

The BDA source code is available for licensing; if interested, please contact Dr. Papamichael at K_Papamichael@lbl.gov.
To learn more about the BDA software and to download a copy of the latest public version (BDA 2.0), please visit http://gaia.lbl.gov/BDA
For Beta Testing of BDA 3.0, please contact Vineeta Pal at VPal@lbl.gov.

DOE-2 DOE-2 DOE-2

PC Version of DOE-2.1E from ESTSC

DOE-2.1E (version 110) for Windows is available from the Energy Science and Technology Software Center (ESTSC). Previously, ESTSC licensed only UNIX and VAX versions. This updated version of DOE-2 incorporates bug fixes and new features such as a Cooled Beam HVAC system and polygon input for walls, floors and ceilings. Like previous DOE-2.1E products from ESTSC, this version accepts textual BDL input but does not have a graphical user interface. Cost of DOE-2.1E-WIN (Version 110) is:

$ 300 U.S. Government, non-profit Educational
$ 575 U.S., Mexico, Canada
$ 1075 Other Foreign

Ed Kidd
NCI Information Systems, Inc.
Energy Science and Technology Software Center
P.O. Box 1020
Oak Ridge, TN 37831

Phone: 865/576-1037
Fax: 865/576-6436
Email: estsc@adonis.osti.gov

DOE-2.1E Documentation on a CD

Most of the DOE-2.1E documentation (including the Engineers Manual, version 2.1A) has been scanned and put on one CD, available for US$100 from ESTSC. Call Ed Kidd to order.
DOE-2.1E Basics and the DOE-2.1E Sample Run Book are not included on the CD; they may be ordered from the National Technical Information Service; go to http://SimulationResearch. lbl.gov >DOE-2 > Documentation.

DOE-2.1E Documentation Updates Free of Charge

Three update documents, in pdf format, are available on our website
http://SimulationResearch. lbl.gov > DOE-2 > Documentation.
The updates are ***not*** cumulative; each document contains different information so you need to download all the packages in order to completely update the existing documentation.

DOE-2 Help Desk

Due to health problems, our regular consultant, Bruce Birdsall, is temporarily unavailable.
In the meantime, please contact the Simulation Research Group with your questions:
Phone: (510) 486-5711, Fax: (510) 486-4089, Email: klellington@lbl.gov

DOE-2 Training

DOE-2 courses for beginning and advanced users:
phone Marlin Addison at (602) 968-2040, or send email to marlin.addison@doe2.com

DOE-2 DOE-2 DOE-2

The Building Energy Simulation User News is published bi-monthly and distributed electronically by the Simulation Research Group at Lawrence Berkeley National Laboratory, with cooperation from the Building Systems Laboratory at the University of Illinois. Direct comments or submissions to Kathy Ellington (KLEllington@lbl.gov). Direct BLAST-related inquiries to the Building Systems Laboratory (support@blast.bso.uiuc.edu). ***© 2001 Regents of the University of California, Lawrence Berkeley National Laboratory. This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology, State and Community Programs, Office of Building Systems of the U.S. Dept. of Energy, under Contract No. DE-AC03-76SF00098***

Software Available from Lawrence Berkeley National Laboratory

Free Downloads	
***BDA 3.0 (Building Design Advisor**) beta*	kmp.lbl.gov/BDA
COMIS (multi-zone air flow and contaminant transport model)	www-epb.lbl.gov/comis
***EnergyPlus* 1.0** (new-generation whole-building energy analysis program, based on BLAST and DOE-2)	SimulationResearch.lbl.gov > EnergyPlus
***GenOpt*®1.1** (generic optimization program)	SimulationResearch.lbl.gov > GenOpt
RADIANCE (analysis and visualization of lighting in design)	radsite.lbl.gov/radiance/
Desktop Radiance (integrates the Radiance Synthetic Imaging System with AutoCAD Release 14)	radsite.lbl.gov/deskrad/
RESEM (Retrofit Energy Savings Estimation Model) (calculates long-term energy savings directly from actual utility data)	eetd.lbl.gov/btp/resem.htm
SUPERLITE (calculates illuminance distribution for room geometries)	eetd.lbl.gov/btp/superlite20.html
THERM 2.1a (model two-dimensional heat-transfer effects in building components where thermal bridges are of concern)	windows.lbl.gov/software/therm/therm.html
WINDOW 5 Beta (thermal analysis of window products)	windows.lbl.gov/software/window/window.html
Request by Fax from 510.486.4089	
RESFEN 3.1 (choose energy-efficient, cost-effective windows for a given residential application)	windows.lbl.gov/software/resfen/resfen.html
Web Based	
Home Energy Saver (quickly compute home energy use)	hes.lbl.gov
Purchase	
SPARK (Simulation Problem Analysis and Research Kernel) (build simulations of innovative building envelope and HVAC systems by connecting component models)	For Windows, SUN, Linux, go to SimulationResearch.lbl.gov > SPARK
ADELINE 2.0 (daylighting performance in complex spaces)	radsite.lbl.gov/adeline/

BLAST*news*

www.bso.uiuc.edu

Building Systems Laboratory (BSL)
30 Mechanical Engineering Building
University of Illinois
1206 West Green Street
Urbana, IL 61801
Telephone: (217) 333-3977
Fax: (217) 244-6534
support@blast.bso.uiuc.edu

The **Building Loads Analysis and System Thermodynamics (BLAST** program predicts energy consumption, energy system performance and cost for new or existing (pre-retrofit) buildings.

BLAST contains three major sub-programs:

- **Space Load Prediction** computes hourly space loads in a building based on weather data and user inputs detailing the building construction and operation.
- **Air Distribution System Simulation** uses the computed space loads, weather data, and user inputs.
- **Central Plant Simulation** computes monthly and annual fuel and electrical power consumption.

Heat Balance Loads Calculator (HBLC)
The BLAST graphical interface (HBLC) is a Windows-based interactive program for producing BLAST input files. You can download a demo version of HBLC (for MS Windows) from the BLAST web site (User manual included).

HBLC/BLAST Training Courses
Experience with the HBLC and the BLAST family of programs has shown that new users can benefit from a session of structured training with the software. The Building Systems Laboratory offers such training courses on an as needed basis typically at our offices in Urbana, Illinois.

WINLCCID 98
LCCID (Life Cycle Cost in Design) was developed to perform Life Cycle Cost Analyses (LCCA) for the Department of Defense and their contractors.

To order BLAST-related products, contact the Building Systems Laboratory at the address above.

Program Name	Order Number	Price
PC BLAST Includes: BLAST, HBLC, BTEXT, WIFE, CHILLER, Report Writer, Report Writer File Generator, Comfort Report program, Weather File Reporting Program, Control Profile Macros for Lotus or Symphony, and the Design Week Program. The package is on a single CD-ROM and includes soft copies of the BLAST Manual, 65 technical articles and theses related to BLAST, nearly 400 processed weather files with a browsing engine, and complete source code for BLAST, HBLC, etc. Requires an IBM PC 486/Pentium II or compatible running MS Windows 95/98/NT.	3B486E3-0898	$1500
PC BLAST Package Upgrade from level 295+	4B486E3-0898	$450
WINLCCID 98: executable version for 386/486/Pentium	3LCC3-0898	$295
WINLCCID 98: update from WINLCCID 97	4LCC3-0898	$195

The last four digits of the catalog number indicate the month and year the item was released or published. This will enable you to see if you have the most recent version. All software will be shipped on 3.5" high density floppy disks unless noted otherwise.

www.bso.uiuc.edu

Recent Reports

LBNL-47972

Modeling Windows in EnergyPlus

F. C. Winkelmann
Lawrence Berkeley National Laboratory
Berkeley CA 94720 USA

ABSTRACT
We give an overview of how windows are modeled in the EnergyPlus whole-building energy simulation program. Important features include layer-by-layer input of custom glazing, ability to accept spectral or spectral-averaged glass optical properties, incidence angle-dependent solar and visible transmission and reflection, iterative heat balance solution to determine glass surface temperatures, calculation of frame and divider heat transfer, and modeling of movable interior or exterior shading devices with user-specified controls. Example results of EnergyPlus window calculations are shown.

Printed copies of this report are available from the Simulation Research Group; please fax your requests to 510.486.4089, attention Kathy Ellington. Be sure to include the title and LBNL number.

Alternatively, you may download a pdf version from http://SimulationResearch.lbl.gov > Publications > Reports > All Technical Reports

★★★★★★★★★★

LBNL-47975

Automatic Unit and Property Conversion in *SPARK*

Edward F. Sowell
California State University, Fullerton

Michael A. Moshier
Chapman University

Ender Erdem
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

ABSTRACT
In object-based models, conversion becomes necessary when there is a mismatch among the representations of same physical quantity at the ports of different objects that need to be connected. The simplest example of this is with regard to physical units of measure, such as temperature. A more complex situation, but with the same character, is with regard to fluid flow, where the state can be represented in terms of any pair of independent properties, and the flow rate can be expressed volumetrically or in terms of mass. In HVAC applications, the most common example is moist air where properties and flow rate can be represented by temperature, relative humidity and volumetric flow, or by enthalpy, humidity ratio and mass flow rate, or other combinations. The need to connect objects with such disparate interfaces arises whenever system model developers following different conventions attempt to share models.

Customarily, conversion is done in an *ad hoc* manner, introducing conversion objects here and there in the problem, as needed. A somewhat more structured approach is to rigidly enforce a common set of units among all classes within a particular application-area library; this approach is recommended for the Neutral Model Format (NMF) (Bring, Sahlin et al. 1992). With this approach one can either undertake "hard conversion" of the underlying equations and re-implement the model, or take the easier path of "wrapping" the offending objects in new "macro objects" with needed converters. Neither choice is very attractive. The first is labor intensive, and errors may be introduced into sound code, while the second can lead to many unnecessary conversion, complicating the model and compromising run-time efficiency.

In this paper we show a unified approach to automatic interface matching that does not suffer the above disadvantages. Special automatic conversion links, or "autolinks," are introduced for this purpose. An autolink carries all admissible representations of one or more variables plus a set of conversion objects that represent constraints that need to be enforced among them. Herein we show that this approach is easily implemented, makes use of existing models with diverse units or properties at the interface, and produces optimum run-time efficiency.

The idea of autolinks has been mentioned before in the literature. Kolsaker and Sahlin make essentially the same suggestion (Sahlin, Bring et al. 1995), but limit their discussion to "property links" only. Sowell and Moshier independently develop the concept, discussing it in terms of unit matching as well as fluid flow and

continued on next page

Recent Reports

properties. They also introduce the idea of automatic elimination of unneeded conversion objects before run-time by "pruning" the computation graph (Sowell and Moshier 1995). In the following, we expand upon the previous work, using examples to clarify the approach, and show the first implementation in the context of the current features of the Simulation Problem Analysis and Research Kernel (SPARK). We then suggest future extensions to made problem specification with autolinks more intuitive.

Printed copies of this report are available from the Simulation Research Group; please fax your requests to 510.486.4089, attention Kathy Ellington. Be sure to include the title and LBNL number.

Alternatively, you may download a pdf version from http://SimulationResearch.lbl.gov > Publications > Reports > All Technical Reports

☆☆☆☆☆☆☆☆☆☆

Modularization and Simulation Techniques for Heat Balance Based Energy and Load Calculation Programs: The Experience of the ASHRAE Loads Toolkit and EnergyPlus

Richard K. Strand
School of Architecture, Univ. of Illinois
Champaign, IL 61820

Curtis O. Pedersen
University of Illinois
Urbana, IL 61801

ABSTRACT
Through sponsorship of the Loads Toolkit and coming changes to the Handbook of Fundamentals, ASHRAE has taken the lead in promoting a heat balance-based approach as the "preferred" method for thermal load and energy analysis calculations. Building on previous ASHRAE research and, to some extent, the BLAST (Building Loads Analysis and System Thermodynamics) program, one of the goals of the Loads Toolkit research project is to obtain a heat balance-based load calculation procedure that is relatively simple in structure where various algorithms, such as different exterior convection coefficient calculation techniques among many others, can be hooked into the heat balance without any restructuring. One of the keys to achieving this goal is the adaptation of legacy versions of a heat balance based approach and their modularization using a modern programming language such as Fortran 90. This process was not a trivial task, and the insight gained in this re-engineering process in a small-scale (single zone) environment provided ideas for modularizing a larger-scale (multiple zone) program such as EnergyPlus. This paper gives an overview of the challenges faced in modularizing the heat balance algorithms in both the Loads Toolkit and EnergyPlus. In addition, it provides an analysis of the resulting heat balance routines in each project and suggestions for the developers of other simulation programs as well as those interested in working with the Loads Toolkit and EnergyPlus.

This paper will be presented at the "Building Simulation 2001" Conference, Rio de Janeiro, August 13-15, 2001, and will be published in the Proceedings.

☆☆☆☆☆☆☆☆☆☆

EnergyPlus: New Capabilities in a Whole-Building Energy Simulation Program

Drury B. Crawley
US Department of Energy
Washington, DC 20585 USA

Linda K. Lawrie
US Army Construction Engineering Research Laboratory
Champaign, Illinois 61821

Frederick C. Winkelmann
Lawrence Berkeley National Laboratory
Berkeley, California 94720

Curtis O. Pedersen
University of Illinois
Urbana, Illinois 61801

ABSTRACT
A new building energy simulation program developed under support from the US government was released in April 2001. EnergyPlus is based on the most popular features and capabilities of BLAST and DOE-2 but is a completely new program written in Fortran 90. New features include variable time steps, user-configurable modular systems, an integrated heat and mass balance-based zone simulation, multizone airflow, air pollutant transport, moisture transfer in building components, solar photovoltaic simulation—and input and output data structures tailored to facilitate third party module and interface development. EnergyPlus is primarily a simulation engine without a user interface—although user interfaces are under development by the private sector. This paper focuses on the general simulation methods and capabilities of EnergyPlus, contrasting it with those of DOE-2 and BLAST. Plans for future releases of EnergyPlus are also described.

This paper will be presented at the "Building Simulation 2001" Conference, Rio de Janeiro, August 13-15, 2001, and will be published in the Proceedings.

☆☆☆☆☆☆☆☆☆☆

continued on next page

Recent Reports

Testing And Validation Of A New Building Energy Simulation Program

Michael J. Witte, Robert H. Henninger
and Jason Glazer
GARD Analytics, Inc.
Park Ridge, IL 60068 USA

Drury B. Crawley
U.S. Department of Energy
Washington, DC 20585 USA

ABSTRACT

Formal independent testing has been an integral component in the development of EnergyPlus, a new building energy simulation program. Testing to date has included analytical, comparative, sensitivity, range, and empirical tests. Published test suites which include reference results have been applied as much as possible in order to take advantage of the efforts of others to develop well-defined, reproducible tests. The results to date show good agreement with well-established simulation tools such as DOE-2.1E, BLAST, and ESP. Several testing utilities have been developed to help automate the task of assuring that each new version of the software is still performing properly. Selected test results are presented along with lessons learned.

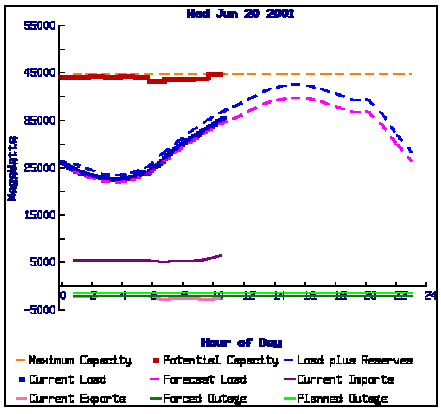
This paper will be presented at the "Building Simulation 2001" Conference, Rio de Janeiro, August 13-15, 2001, and will be published in the Proceedings.

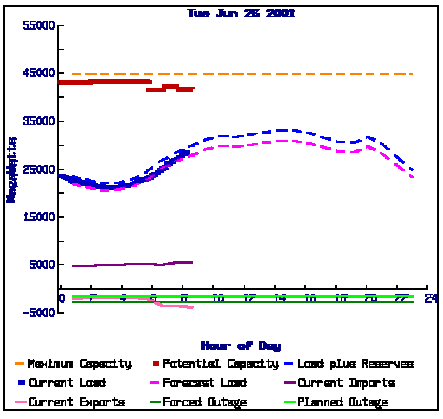
☆☆☆☆☆☆☆☆☆☆

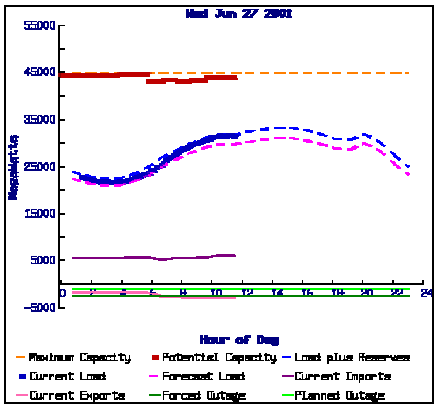
Today's Supply of and Demand for Electricity in California

http://energycrisis.lbl.gov

The power grid that supplies the electric current coming into your California home or business is designed to maintain a dynamic balance between the consumer demand for electricity and the amount being supplied by generators. This web site presents a chart that shows an approximate representation of this dynamic balance. Quantities that are forecasts or estimates are shown by dashed lines. The current load is published every 10 minutes by the California Independent System Operator (ISO) for the area it controls, which covers about 80% of electricity use in the state. It is more difficult to quantify the amount of supply that may be available, which we call "Potential Capacity". Our approximation is based on: the total capacity of generators licensed to operate, minus the generators that are out of service (forced and planned outages), plus imports, minus exports. Outages are updated daily, and imports and exports are updated hourly.

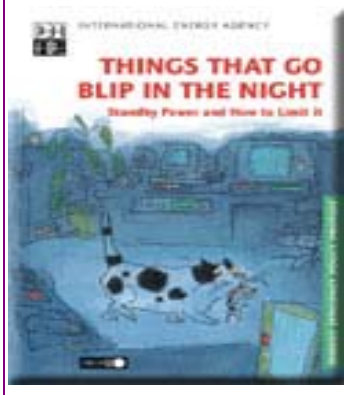






NEW BOOKS

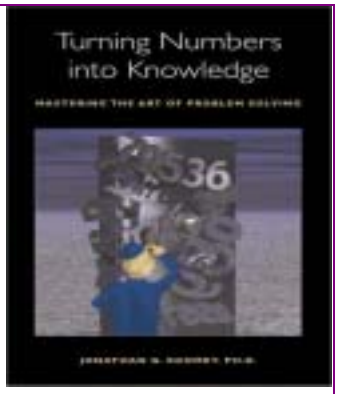
Switched Off But Not Unplugged



A new International Energy Agency (IEA) publication ***Things That Go Blip in the Night: Standby Power and How to Limit It*** examines the problem posed by growing standby power consumption, as more appliances are equipped with remote-control, network-sensing and digital-display features. The book explores ways to reduce this unnecessary energy, including international collaboration, test procedures, standards and voluntary efforts .

Things That Go Blip in the Night: Standby Power and How to Limit It is available from the International Energy Agency, IEA Books
Fax +33 1 40 57 65 59
e-mail BOOKS@IEA.ORG

Turning Numbers into Knowledge



***Turning Numbers Into Knowledge: Mastering the Art of* Problem Solving**

A new book by LBNL's Jonathan G. Koomey (jgkoomey@lbl.gov) teaches you (in a non-technical way) the art of using numbers for practical problem solving, revealing tools, tricks, and heretofore unwritten rules that the best real-world problem solvers know by heart.

Turning Numbers Into Knowledge: Mastering the Art of Problem Solving is available from amazon.com.
For details and excerpts, please visit the book's website at:

www.numbersintoknowledge.com